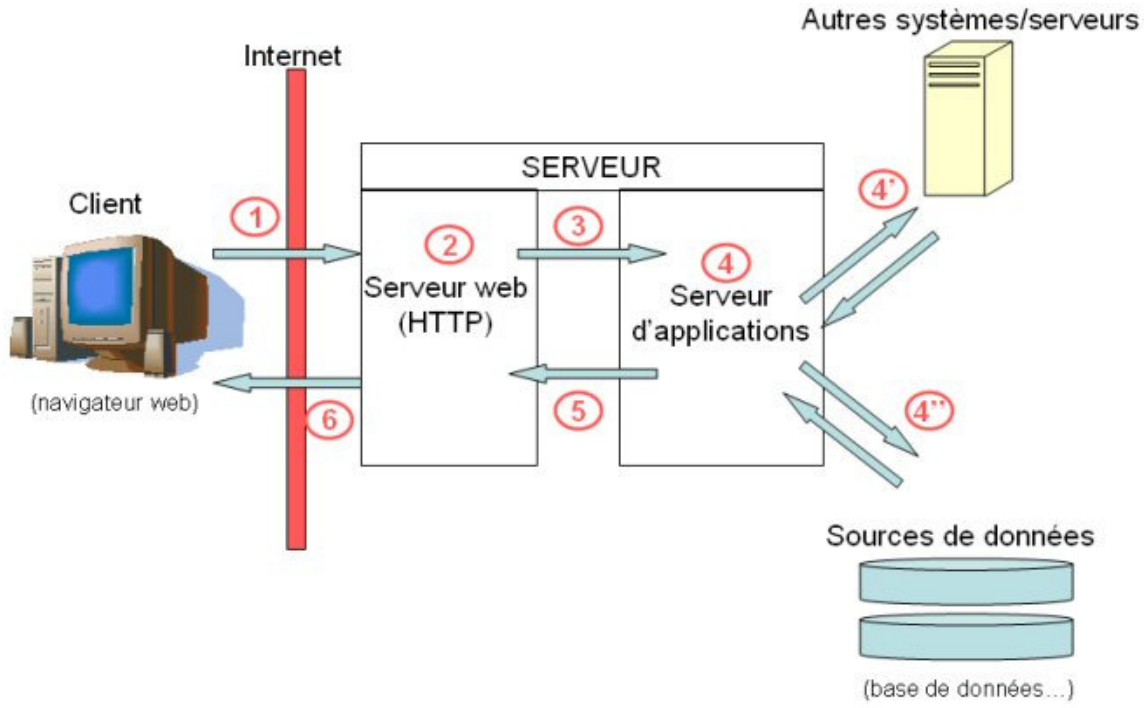
Chapitre 5. Interactivité sur le web : les applets Java

Définition

Une applet est une application Java insérée dans une page web. Ce genre d'application est appelée *application client* car elle est utilisée par celui qui appelle la page web (le client), et non par celui qui la génère (le serveur).

Il ne faut pas confondre un applet et une application J2EE (maintenant JEE). Pour bien comprendre à quel niveau se situe l'applet dans le cycle de vie d'une page web, un petit schéma s'impose. Ce schéma représente le cycle de vie d'une page web dite dynamique, c'est-à-dire que celle-ci contient du code interprété par le serveur (Java, PHP, ASP...) qui est ici une page contenant du code Java :



- (1) **Le client émet une requête** (saisie d'une URL, clic sur un lien...) pour demander une ressource au serveur. Exemple :***http://www.monserveur.com/tuto.do***. Il ne sait pas ici si la réponse qui va lui parvenir est statique (page HTML simple) ou dynamique (générée par une application web). Dans notre cas, il s'agit d'une application répondant à l'adresse "***tuto.do***" sur le serveur "***monserveur.com***".
- (2) Côté serveur, **c'est le serveur web (exemple : Apache) qui traite les requêtes HTTP** entrantes. Il traite donc toutes les requêtes, qu'elles demandent une ressource statique ou dynamique. Seulement, un serveur HTTP ne sait répondre qu'aux requêtes visant des ressources statiques. **Il ne peut que renvoyer des pages HTML, des images, des applets... existantes**.
- (3) Ainsi, si le serveur HTTP s'aperçoit que la requête reçue est destinée au serveur d'applications, il la lui transmet. Les deux serveurs sont reliés par un canal, nommé "***connecteur***".
- (4) **Le serveur d'applications (exemple : Tomcat ! Serveur d'applications Java) reçoit la requête à son tour**. Il est, lui, en mesure de la traiter. Il exécute donc le morceau d'application (la *servlet*) auquel est destinée la requête, en fonction de l'URL. Cette opération est effectuée à partir de la configuration du serveur. La servlet est donc

invoquée, et le serveur lui fournit notamment deux objets Java exploitables : un représentant la requête, l'autre représentant la réponse. **La servlet peut maintenant travailler, et générer la réponse à la demande**. Cela peut passer par la consultation de sources de données, comme des bases de données (4" sur le schéma). Ou bien par l'interrogation d'autres serveurs ou systèmes (4' sur le schéma), l'environnement Java web permettant de se connecter à de nombreux systèmes.

- **(5) Une fois sa réponse générée, le serveur d'applications la renvoie, par le connecteur, au serveur web**. Celui-ci la récupère comme s'il était lui-même allé chercher une ressource statique. Il a simplement délégué la récupération de la réponse, et celle-ci a été générée, mais ce n'est plus le problème.
- **(6) La réponse est dorénavant du simple code HTML**, compréhensible par un navigateur. Le serveur HTTP peut donc retourner la réponse au client. Celle-ci contient toutes les ressources nécessaires (feuilles Javascript, feuilles CSS, applet Java, images...).

Les servlets sont en fait des classes Java. Celles-ci peuvent être des classes Java programmées par un développeur ou une pageJSP (page web contenant du code Java) compilée en servlet par le serveur d'application avant traitement par celui-ci. Nous reviendrons sur tous ces points plus tard, nous n'avons pas besoin d'en savoir plus pour le moment.

Ceci est un résumé du cycle de vie d'une page web faite avec la technologie J2EE. Je me doute que vous devez avoir des sueurs froides, mais ne vous inquiétez pas, nous reverrons tout ça plus en détail lorsque nous aborderons le développement web en Java...
Pour le moment, tout ce que vous avez besoin de savoir c'est qu'**un applet est une ressource utilisée par votre navigateur, tout comme une image : à la différence que là, il s'agit d'un programme qui va s'exécuter sur votre page !**

Les ressources utilisées par votre navigateur pour charger et utiliser une applet sont chargées au chargement de la page, après que le serveur web ait renvoyé la réponse à votre requête.
Ces ressources sont dans le code source HTML de la page et le navigateur charge tout ce dont il a besoin pour afficher la page comme le développeur l'a souhaité (images, feuilles CSS, applet...).

Attention : il se peut que votre navigateur n'autorise pas l'exécution des applets Java !
Pour remédier à ce problème, vous devez allez dans les options internet : ***menu Outils > Options*** dans l'onglet **Contenu** : cochez "***autoriser le Java***", sous Firefox.
Sous IE 7, faites : ***Outils > Options internet***, dans l'onglet "***content***", cochez la case "***utiliser JRE 1.X.XX pour les applets***" où X.XX correspond à la version de votre JRE installé sur votre machine.

Maintenant, vous savez distinguer une application client d'une application serveur et donc, vous ne devrez plus faire l'amalgame entre applet et servlet !

Votre première applet

Il est temps maintenant de faire votre première applet.
Vous allez voir que c'est très simple et que ça ressemble beaucoup à ce que vous avez fait jusque-là. En fait, c'est quasiment identique à une exception près :
un applet n'a pas de constructeur mais elle utilise la méthode init() de la super-classe `Applet` du package `awt` ou`JApplet` du package `swing`.

Codage de l'applet

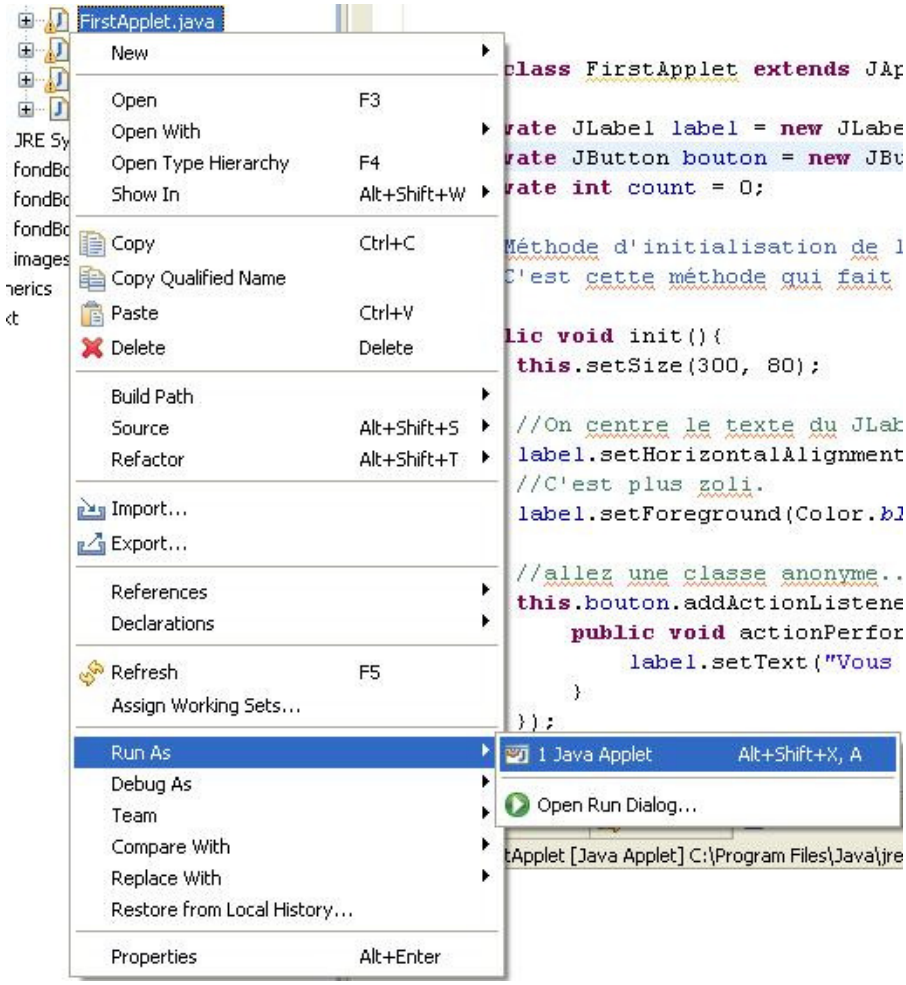
Nous allons faire un applet avec un code minimal, disons un label et un bouton. Lors du clic sur `bouton`, nous afficherons le nombre de clics effectués. Rien de bien méchant. Créez un nouveau projet avec une classe **`FirstApplet`** héritée de **`JApplet`**.
Voici le code source de votre première applet :

Code : Java

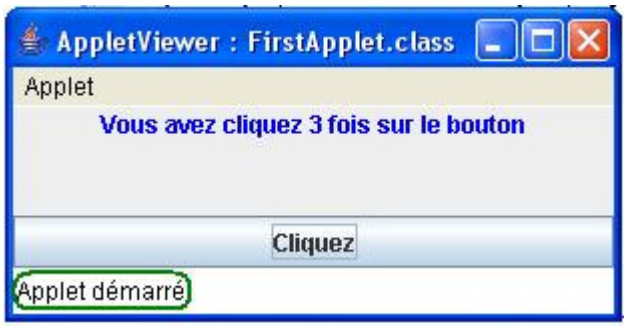
```
1  import java.awt.BorderLayout;
2  import java.awt.Color;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5
6  import javax.swing.JApplet;
7  import javax.swing.JButton;
8  import javax.swing.JLabel;
9
10
11 public class FirstApplet extends JApplet {
12
13         private JLabel label = new JLabel();
14         private JButton bouton = new JButton("Cliquez");
15         private int count = 0;
16         /**
17          * Méthode d'initialisation de l'applet
18          * C'est cette méthode qui fait office de constructeur
19          */
20         public void init(){
21                 this.setSize(300, 80);
22
23                 //On centre le texte du JLabel et on écrit en bleu...
24                 label.setHorizontalAlignment(JLabel.CENTER);
25                 //C'est plus zoli.
26                 label.setForeground(Color.blue);
27
28                 //Allez, une classe anonyme... Just for the fun ;)
29                 this.bouton.addActionListener(new ActionListener(){
30                         public void actionPerformed(ActionEvent arg0) {
31                                 label.setText("Vous avez cliqué " +
32 (++count) + " fois sur le bouton");
33                         }
34                 });
35
36                 //On ajoute nos composants
37                 this.getContentPane().add(bouton, BorderLayout.SOUTH);
38                 this.getContentPane().add(label, BorderLayout.NORTH);
39                 //Et le tour est joué !
40         }
41
42 }
```

Avant de vous lancer dans le test de cette première applet, vous devez savoir tout de même que, mis à part quelques méthodes comme setTitle("Animation") , setLocationRelativeTo(null) et quelques autres propres aux objets **JFrame**, les applets s'utilisent de la même manière. Bien sûr, avec la méthode public void init() à la place d'un constructeur !

Maintenant, avant d'intégrer votre programme dans une page web, vous devez vous assurer que celui-ci fonctionne correctement. Comment on fait ça ? Tu nous a dit que les applets étaient utilisées dans des pages web. Je maintiens, mais Eclipse vous propose un moyen d'exécuter votre classe comme un applet. Pour faire cela, faites un clic droit sur votre fichier puis choisissez "**Run as**" et enfin choisissez "**Java Applet**", comme ceci :



Vous pouvez voir, ébahis, le résultat de votre applet :



Vous avez un indicateur vous signalant si votre applet est lancé ou non. Si vous voyez le message : "***Applet not inited***", ça veut dire qu'une erreur s'est glissée dans votre code et que la JVM n'a pas pu initialiser l'applet !

Insertion dans une page HTML

Pour que votre navigateur sache que la ressource à utiliser est un applet Java, vous devez utiliser la balise HTML<applet></applet> .
Celle-ci peut prendre plusieurs attributs et vous pouvez même passer des paramètres à votre applet grâce à cette balise.
Voici la liste des paramètres que peut prendre la balise <applet></applet> :

- name="FirstAnimation" : nom de l'applet, ici FirstAnimation . Nous allons vite voir l'intérêt de cet attribut ;
- width="300px" : largeur de l'applet affiché, ici, 300 pixels ;
- height="300px" : hauteur de l'applet affiché ;
- codebase="class/" : l'URL de base pour l'applet, c'est-à-dire l'endroit où le navigateur peut trouver les fichiers .class ; ici, les fichiers sont dans le dossier ***class*** à côté de votre fichier HTML ;
- code="FirstAnimation.class" : fichier de classe de l'applet. Celui où se trouve la méthode init() ;
- archive="plugin.jar" : identifie les ressources à pré-charger (.jar, images...) ; ici, nous pré-chargeons une archive Java appelée plugin.jar ;
- alt="Please Wait..." : affiche un texte au cours du chargement ;
- hspace="10px" : espacement horizontal entre l'applet et un autre contenu (div ou autre bloc HTML...) ;
- vspace="20px" : idem que précédemment mais pour l'espacement vertical.

Il y en a des attributs pour cette balise ? Voici donc un exemple de balise applet :

Code : HTML

```
<applet name="FirstAnimation" codebase="class/"
code="FirstAnimation.class"
        height="300px" width="300px" archive="plugin.jar">
   <param name="message" value="Message pour les ZérOs">
</applet>
```

On comprend bien tout mais, qu'est-ce que c'est que ce truc : <param name="message" value="Message pour les ZérOs"> ?

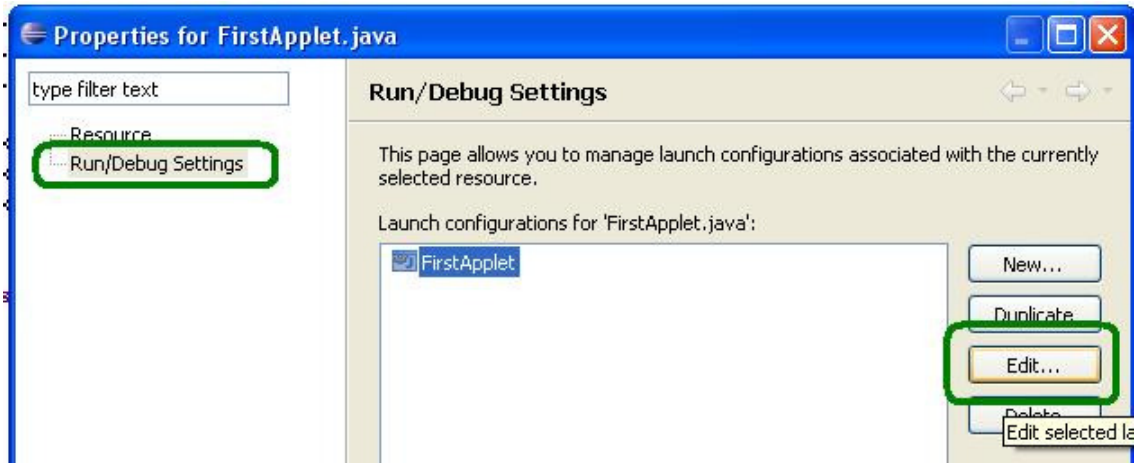
Je vous ai dit que vous pouviez passer des paramètres à votre applet. Eh bien c'est comme ceci que nous allons nous y prendre ! Ceci veut dire que nous pourrons utiliser la méthode getParameter(String paramName); qui va nous renvoyer un `String`correspondant à l'attribut `value` de la balise. Ici, on aurait this.getParameter("message")//Retourne : Message pour les ZérOs.

Tenez, nous allons essayer ça ! Ajoutez ceci dans votre méthode init() : System.out.println("Paramètre passé via la balise <param> : " + this.getParameter("message")); .

Lancez votre applet et :

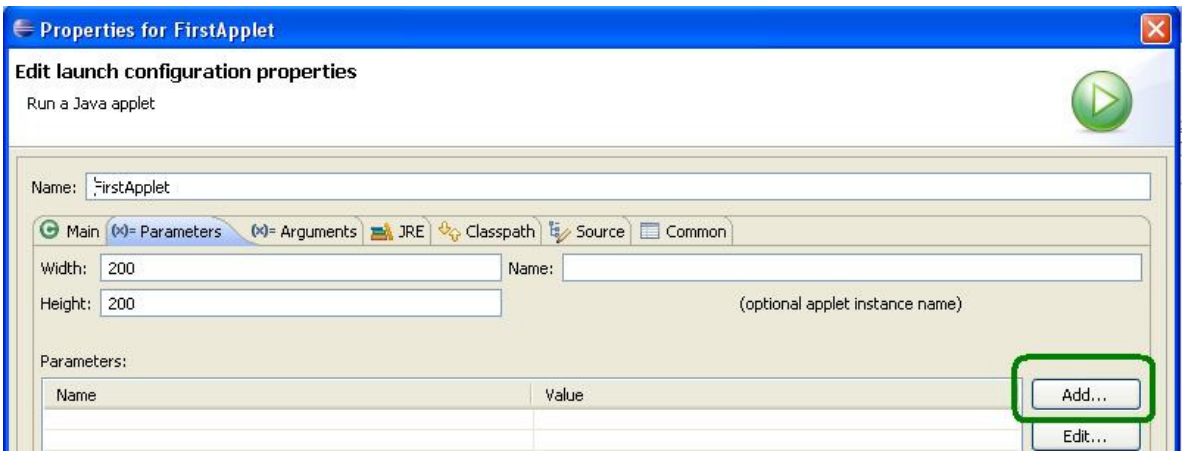
Ah oui ! Si vous ne spécifiez pas de paramètre pour votre applet, ledit paramètre vaut **null**.

Voici comment on procède pour spécifier un paramètre pour votre application. Déjà, faites un clic droit sur votre fichier puis allez dans **Propriétés**.
Ensuite, cliquez sur "**Run/Debug settings**" puis sur le fichier correspondant à votre applet et enfin sur "**Edit**", comme ceci :

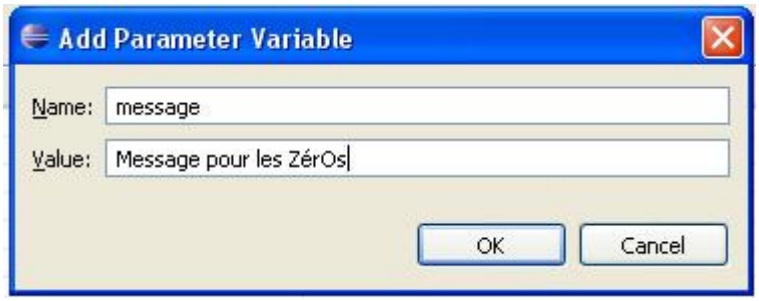


Dans la fenêtre dans laquelle vous êtes maintenant, choisissez l'onglet "**parameter**".

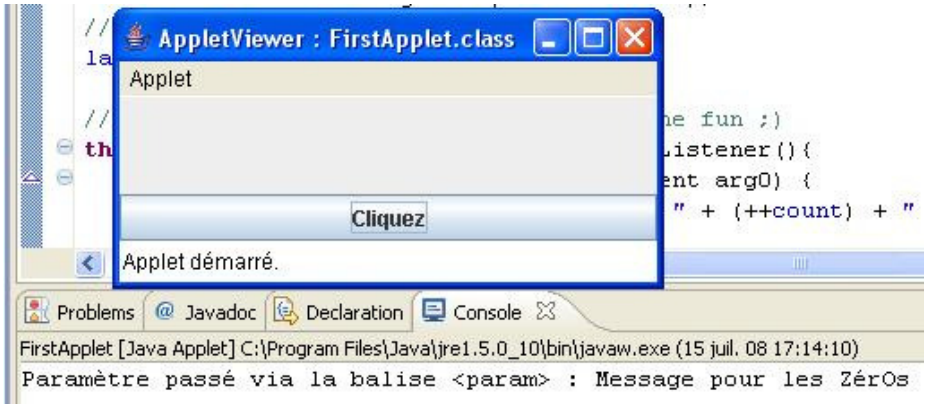
Vous arrivez enfin à l'endroit où vous pouvez ajouter des paramètres. Cliquez sur "**Add**", comme ceci :



Et enfin, renseignez le nom de votre paramètre ainsi que sa valeur :



Cliquez sur "**Ok**" sur toutes les fenêtres que vous avez ouvertes et relancez votre applet, le paramètre s'affiche enfin !



Vous pouvez maintenant incorporer votre applet dans une page HTML.
Voici le code de ladite page :

Code : HTML

```
<html>

   <body style="margin:auto;">

        <div style="width:320px;margin:auto;margin-top:100px;border:5px solid black">
                <applet codebase="class/" code="FirstApplet.class" height="80" width="300" hspace="10" vspace="10">
                        <param name="message" value="Message pour les ZérOs">
                </applet>
        </div>

   </body>
</html>
```

J'ai créé ce code et enregistré le fichier contenant ce code HTML sur mon bureau, j'ai donc dû déplacer mes fichiers .class - oui, vous ne rêvez pas, j'ai dit : **mes fichiers .class** - dans un dossier, que j'ai appelé "***class***" pour l'occasion (cf. paramètre codebase de l'applet)...

J'ai récupéré mes fichiers .class dans le répertoire ***/bin*** de votre projet, et vous pouvez voir que vous avez **FirstApplet.class** et**FirstApplet$1.class** dans le cas où vous avez exactement le même code que moi.

À quoi ça correspond ? En fait, **FirstApplet.class** correspond à la compilation de votre classe **`FirstApplet`** et **FirstApplet$1.class** correspond à la compilation de votre classe anonyme ! Pas de nom pour cette classe, donc la JVM remplace le nom par "***$1***". Si vous aviez utilisé une classe

interne, appelons-la `BoutonListener` et si vous compiliez le code, vous auriez toujours **FirstApplet.class**, mais vous auriez eu le fichier **FirstApplet$BoutonListener.class**.

Donc, si vous avez créé votre page web ailleurs que dans votre dossier contenant votre projet, vous devrez déplacer tous les fichiers .class commençant par **`FirstApplet`** et toutes les autres ressources que votre applet utilise (images, autres classes, archives Java...).
Vous pouviez aussi créer votre page web dans le dossier de votre projet et spécifier comme codebase "**`bin/`**", dossier contenant vos .class dans le projet d'Eclipse... C'est à votre bon vouloir !

Il faut que vous sachiez que, si Eclipse est assez laxiste pour lancer l'applet même si le paramètre "**`message`**" n'est pas renseigné, votre navigateur, enfin la JVM de votre navigateur sera moins conciliante... **Si le paramètre est manquant, l'applet plantera !** Voilà, vous venez de faire votre première applet ! Alors, heureux ?

Nota Bene

Avant de continuer, vous devez savoir une dernière chose, ceci ne concerne pas directement Java mais la balise<applet></applet> .
En fait, depuis la sortie de HTML 4.0, la balise <applet></applet> est dépréciée par le W3C, c'est-à-dire que cet organisme préconise l'utilisation de la balise <object></object> .

Ceci en grande partie à cause de IE qui gérait le Java avec sa propre JVM (version 1.1, c'est vieux...) et non celle de Sun Microsystems (bientôt 1.7...). Il faut, afin que la balise <applet></applet> fonctionne correctement sous IE, avoir installé un environnement Java et s'assurer que IE utilise celui-ci pour interpréter du Java... (cf. plus haut).

Je ne détaillerai pas l'utilisation de cette balise vu que Sun Microsystems recommande l'utilisation de la balise <applet></applet>... Voici tout de même un exemple d'utilisation de la balise <object></object> :

Code : HTML

```
<!-- L'Utilisation des commentaires conditionnels propres à IE sont à utiliser -->
<!-- car même si IE requiert l'utilisation de cette balise, il ne l'interprète pas comme les autres -->
<!--[if IE]>
<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93" width="300" height="80" name="FirstApplet">
   <param name="java_code" value="FirstApplet.class" />
   <param name="java_codebase" value="class/" />
   <param name="type" value="application/x-java-applet;version=1.5" />

<![endif]-->
<p>
<object classid="java:FirstApplet.class"
          codebase="class"
          type="application/x-java-applet"
          width="300" height="80">
    <param name="code" value="FirstApplet" />
    <!-- Safari a besoin de ces paramètres -->
    <param name="JAVA_CODEBASE" value="class" />
    <param name="BGCOLOR" value="000000" />
    <param name="TEXTCOLOR" value="FF0000" />
    <param name="TEXT" value="Test :-)" />
    <param name="SPEED" value="250" />
    <param name="RANDOMCOLOR" value="1" />
    alt : <a href="class/FirstApplet.class">FirstApplet.class</a>
  </object>
</p>
```

Il est grand temps de voir comment notre page web peut interagir avec notre applet.
Go !
Avant de terminer ce chapitre, vous devez savoir que les applets n'ont pas le droit de tout faire sur une page web !

Applets et sécurité

En fait, au même titre que Javascript, les applets Java n'ont pas le droit d'accéder à la machine du client. Pour faire simple, ils sont confinés dans le navigateur web.
Et heureusement ! Vous imaginez un peu toutes les dérives si ce genre de script ou de programme pouvait naturellement avoir accès à votre PC ? Là, on pourrait devenir parano et il vaudrait mieux !

- Est-ce que ce script n'est pas en train d'effacer un fichier de configuration ?
- Il me semble que cette applet est en train d'accéder à un répertoire sensible...
- ...

Bref, vous imaginez.

Cependant, il se peut que quelquefois, pour quelques rares cas, votre applet doive accéder à des ressources de votre PC.

Exemple

Dans la boîte dans laquelle je suis actuellement, nous sommes en train de développer une application, format client léger (web), afin de gérer tous les processus industriels de la société, dont la pesée de certains articles avec scan des documents en même temps.
Nous avons donc fait un applet qui s'occupe de faire tout ça mais pour communiquer avec les ports COM et le scanner, nous avons dû **signer notre applet**.

Comment ça signez l'applet ?
Nous avons signé notre applet, c'est-à-dire que nous avons créé un certificat que nous avons attribué à notre applet et que l'utilisateur DOIT soit accepter, soit refuser au chargement de la page : ce certificat stipule que l'applet peut accéder à des ressources de sa machine, et lui demande s'il veut lui faire confiance .
Il n'est pas très pertinent de parler de la façon de signer une applet : au pire, si vous avez vraiment besoin de ça, Google est votre ami.
Vous savez tout de même que les applets n'ont pas tous les droits sur une page web, au même titre que Javascript. Vous avez vu pas mal de choses, mine de rien, dans ce chapitre.

Ce que vous devez retenir

- Les applets peuvent dériver de `java.awt.Applet` ou de `javax.swing.JApplet`.
- Les applets n'ont pas de constructeur mais une méthode `init()`.
- En gros, les applets se comportent comme des applications fenêtrées.
- Par défaut, **les applets n'ont pas accès aux ressources de la machine client**.
- Pour accéder à la machine du client, **vous devrez signer votre applet !**

Référence :

Pour plus de détail, consulter ce tutoriel sur :
http://www.siteduzero.com/tutoriel-3-10564-les-applets.html